

2/PRTS

09/646796

529 Rec'd PCT/PTO 22 SEP 2000

1 APPARATUS AND METHOD FOR PROVIDING TRANSACTION SERVICES.

2

3 The present invention relates to apparatus and a method  
4 for providing transaction services. In particular it  
5 relates to networked computer-based transaction machines  
6 and a method for providing transaction services using  
7 said transaction machines.

8

9 Transaction machines are herein defined as any computer-  
10 based machine able to interact with a user.

11

12 The term ATM is used herein to refer to any transaction  
13 machine able to dispense cash. Typically, such machines  
14 can also undertake physical transactions such as  
15 inputting information through a keypad or touch screen,  
16 making sounds, producing video and printing. They might  
17 also be able to read bank cards and such like. Kiosks  
18 are transaction machines unable to dispense cash, but  
19 otherwise able to provide a range of interactive  
20 features, often relating to financial services. For test  
21 purposes, a conventional PC may be used as a transaction  
22 machine.

23

24 Electronic cash machines are a large and rapidly growing  
25 market. Many different hardware providers produce

1 equipment for this market such as the machines  
2 themselves, the servers to which they connect and the  
3 networking means through which they typically  
4 communicate. Furthermore, many different operating  
5 systems and applications are used both for operating and  
6 developing these systems.

7  
8 As a result of the complexity and diversity of hardware  
9 and software currently being used in this field, it is  
10 difficult and expensive to alter these systems to extend  
11 their functionality, upgrade to newer and better  
12 hardware, software or networking means or to interface  
13 with other systems. As it is difficult to make even  
14 small changes to complex systems without running the risk  
15 of their malfunctioning, the evolution of such systems is  
16 slow.

17  
18 It would therefore be advantageous to find a way of  
19 making it easier to alter the hardware, software and  
20 network components of ATMs/kiosks, their servers and  
21 their networking means.

22  
23 Furthermore, it would be advantageous to provide a means  
24 for enabling such changes to be implemented in small  
25 stages.

26  
27 Yet further, it would be advantageous to find a way to  
28 reduce the risk of such systems malfunctioning.

29  
30 In current practice, it is difficult and therefore  
31 expensive to operate ATM/kiosk networks containing  
32 diverse hardware, software and networking means. Often  
33 large amounts of hardware and software must be upgraded  
34 concomitantly to reduce interface problems. Furthermore,  
35 it is difficult to interface networks of dissimilar

1 devices, perhaps belonging to different organisations.  
2 If dissimilar ATM/kiosk systems could be readily  
3 interfaced, forming a so-called Extranet, new and useful  
4 co-operative applications could be developed which,  
5 although currently possible, are prohibitively complex  
6 and expensive at the present time.

7  
8 It would therefore be advantageous to provide a better  
9 means of networking ATMs/kiosks which use diverse  
10 hardware, software and networking implementations. In  
11 particular, it would be advantageous to provide a means  
12 of allowing co-operation between dissimilar networks.  
13 Furthermore, it would be advantageous to reduce the  
14 amount of work required to enable ATM/kiosk applications  
15 to run on dissimilar hardware implementations.

16  
17 At the present time, there is a rapid growth in  
18 electronic commerce (e-commerce), usually conducted over  
19 the internet. E-commerce is being limited by  
20 difficulties gaining access to the internet for many  
21 consumers and due to the limitations of the machines  
22 currently used by consumers for internet transactions. A  
23 typical e-commerce consumer will access a web site using  
24 a home PC. However, home PCs lack facilities such as the  
25 ability to dispense cash or read a smartcard which are  
26 important in many types of common financial transaction.

27  
28 It would therefore be desirable to provide a means of  
29 allowing internet-based e-commerce to be accessed from  
30 ATMs and kiosks which already have hardware facilities  
31 suitable for financial transactions. This would allow e-  
32 commerce services to be provided which required expensive  
33 or high-security hardware facilities which cannot be  
34 securely provided at a reasonable cost on privately owned  
35 web browsers. Furthermore, it would be possible for e-

1 commerce to be made readily available to a much larger  
2 base of consumers than is currently available.

3  
4 The design of ATM networks typically involves input from  
5 numerous professionals such as software and hardware  
6 engineers specialising in the various systems,  
7 applications and communications means, graphics and GUI  
8 specialists, language specialists and so forth. In  
9 current working practice these specialists are highly  
10 dependent on each other and much time and money is spent  
11 communicating different requirements amongst people  
12 working on diverse areas of a project.

13  
14 It would therefore be advantageous to provide a means by  
15 which the different specialists working on a project may  
16 work more independently. In particular, it would be  
17 highly advantageous to provide a means by which the  
18 different specialists may customise elements of the  
19 application pertaining to their own specialisation  
20 without affecting other elements of the application. It  
21 would be particularly advantageous if the different  
22 specialists were able to use well known prior art  
23 authoring tools to prepare aspects of the application.  
24 According to the present invention there is provided a  
25 method for providing transaction services wherein

26  
27 (a) the user of the transaction services interacts  
28 with a computer-based transaction machine which is  
29 controlled by one or more software applications;

30  
31 (b) the software applications interact with the  
32 functional interfaces of middleware software, which  
33 extends the functionality of an underlying operating  
34 system; and

35

1 (c) said functional interfaces provide functionality  
2 which is implemented in a manner adapted to the  
3 particular hardware capabilities of the transaction  
4 machine.

5

6 The computer-based transaction machine may be selected  
7 from a group which comprises automatic teller machines,  
8 kiosks, electronic point of sale machines and the like.

9

10 Preferably, the middleware software comprises a series of  
11 transaction objects and controls for standard device  
12 functions.

13

14 More preferably, transaction objects are independent of  
15 the interface between the user and the transaction  
16 machine; the interface between the user and the  
17 transaction machine being customisable.

18

19 Preferably, the controls implement a capabilities  
20 interface.

21

22 More preferably, the capabilities interface is able to  
23 communicate the capabilities of the control software.

24

25 The applications, objects and controls may be fully  
26 concurrent and asynchronous.

27

28 The controls may have a mode in which events are queued  
29 up and delivered to the application on demand.

30

31 Preferably, controls can run on the transaction machine  
32 even when supported hardware devices are not present.

33

1 More preferably, the middleware software uses one or more  
2 open standards for interacting with different hardware  
3 systems.

4  
5 Preferably, the middleware software only provides  
6 cancellation commands for functions which can be  
7 successfully cancelled.

8  
9 The middleware software may only requires a timeout  
10 command to be supplied when it is meaningful to do so.

11  
12 Preferably, all controls are persistent.

13  
14 More preferably, there is provided a control containing a  
15 persistent object.

16  
17 Preferably, all errors and transgressions are asserted by  
18 the middleware software.

19  
20 Preferably, the middleware software provides a trace  
21 facility that is always enabled and which logs trace  
22 events.

23  
24 The middleware software may use a ring buffer to store a  
25 log of trace events.

26  
27 Preferably, the middleware software writes trace data to  
28 memory and then copies it to disk only when the  
29 transaction machine is idle.

30  
31 Preferably, one or more software applications are hosted  
32 in a web browser.

33

1 More preferably, the use of a web browser provides  
2 support for software distribution and network  
3 connections.

4  
5 An additional browser frame may be provided which  
6 contains the device controls required to detect events  
7 which must be dealt with immediately they occur.

8  
9 The middleware software may comprise a series of COM  
10 components with a scriptable ActiveX® interface.

11  
12 The middleware software may comprise a series of  
13 Javabeans™ components with a scriptable interface.

14  
15 The use of a web browser may allow conventional web sites  
16 to be displayed by the computer-based transaction  
17 machine.

18  
19 Preferably, the middleware software allows or disallows  
20 access to particular web sites according to a rule  
21 database.

22  
23 The middleware software may be adapted to customise time-  
24 out of the display of individual internet web sites.

25  
26 Preferably, said computer-based transaction machine is  
27 adapted to allow the software applications and middleware  
28 to be altered across a network by an authority.

29  
30 More preferably, the transaction machine communicates  
31 information about its status to a remote monitoring  
32 station across a network.

33  
34 According to a second aspect of the present invention,  
35 there is provided a computer-based transaction machine;

1 wherein said computer-based transaction machine is  
2 provided with hardware devices for interaction with users  
3 and the exchange of transaction-related information with  
4 other machines; wherein said computer-based transaction  
5 machine is controlled by one or more software  
6 applications; wherein said software applications control  
7 hardware devices through functional interfaces with  
8 middleware software; wherein said middleware software  
9 extends the functionality of an underlying operating  
10 system and wherein said functional interfaces are  
11 hardware independent but provide functionality which is  
12 implemented in a manner adapted to the capabilities of  
13 the particular hardware devices which are provided.

14  
15 The computer-based transaction machine may be selected  
16 from a group which comprises automatic teller machines,  
17 kiosks, electronic point of sale machines and the like.

18  
19 Preferably, the middleware software comprises a series of  
20 transaction objects and controls for standard device  
21 functions.

22  
23 More preferably, transaction objects are independent of  
24 the interface between the user and the transaction  
25 machine; the interface between the user and the  
26 transaction machine being customisable.

27  
28 Preferably, the controls implement a capabilities  
29 interface.

30  
31 More preferably, the capabilities interface is able to  
32 communicate the capabilities of the control software.

33  
34 The applications, objects and controls may be fully  
35 concurrent and asynchronous.



1  
2 The controls may have a mode in which events are queued  
3 up and delivered to the application on demand.  
4  
5 Preferably, controls can run on a transaction machine  
6 even when supported hardware devices are not present.  
7  
8 More preferably, the middleware software uses one or more  
9 open standards for interacting with different hardware  
10 systems.  
11  
12 Preferably, the middleware software only provides  
13 cancellation commands for functions which can be  
14 successfully cancelled.  
15  
16 The middleware software may only requires a timeout  
17 command to be supplied when it is meaningful to do so.  
18  
19 Preferably, all controls are persistent.  
20  
21 More preferably, there is provided a control containing a  
22 persistent object.  
23  
24 Preferably, all errors and transgressions are asserted by  
25 the middleware software.  
26  
27 Preferably, the middleware software provides a trace  
28 facility that is always enabled and which logs trace  
29 events.  
30  
31 The middleware software may use a ring buffer to store a  
32 log of trace events.  
33

1 Preferably, the middleware software writes trace data to  
2 memory and then copies it to disk only when the  
3 transaction machine is idle.

4  
5 Preferably, one or more software applications are hosted  
6 in a web browser.

7  
8 More preferably, the use of a web browser provides  
9 support for software distribution and network  
10 connections.

11  
12 An additional browser frame may be provided which  
13 contains the device controls required to detect events  
14 which must be dealt with immediately they occur.

15  
16 The middleware software may comprise a series of COM  
17 components with a scriptable ActiveX<sup>®</sup> interface.

18  
19 The middleware software may comprise a series of  
20 Javabeans<sup>™</sup> components with a scriptable interface.

21  
22 The use of a web browser may allow conventional web sites  
23 to be displayed by the computer-based transaction  
24 machine.

25  
26 Preferably, the middleware software allows or disallows  
27 access to particular web sites according to a rule  
28 database.

29  
30 The middleware software may be adapted to customise time-  
31 out of the display of individual internet web sites.

32  
33 Preferably, the computer-based transaction machine is  
34 adapted to allow the software applications and middleware  
35 to be altered across a network by an authority.

1

2 More preferably, the transaction machine can communicate  
3 information about their status to a remote monitoring  
4 station across a network.

5

6 According to a third aspect of the present invention  
7 there is provided a network comprising a plurality of  
8 computer-based transaction machines, one or more  
9 networking means and one or more application servers.

10

11 According to a fourth aspect of the present invention,  
12 there is provided an Extranet formed by combining a  
13 plurality of networks of computer-based transaction  
14 machines.

15

16 Preferably, the Extranet is provided with a security  
17 mechanism which limits the hardware functionality  
18 available to individual software applications.

1 An example embodiment of the present invention, referred  
2 to as the system, will now be described with reference to  
3 the following Figures wherein:

4

5 Figure 1 shows a simple ATM network;

6 Figure 2 shows an ATM network with diverse hardware;

7 Figure 3 shows two distinct networks being combined  
8 to form an Extranet; and

9 Figure 4 shows the software architecture of the  
10 preferred implementation of the system.

11

12 Figure 1 shows a simple ATM network comprising a server  
13 1, a networking means 2 and an ATM 3. The system is  
14 designed to operate such networks and also more complex  
15 networks such as shown in Figure 2 wherein there may be  
16 ATMs of different functionality, here labelled 4.

17

18 A particular benefit of the system is its ability to  
19 allow distinct networks to operate together as shown in  
20 Figure 3. Here, two distinct networks 5 and 6 operated  
21 by distinct servers 7 and 8 are connected 9. The  
22 resulting joined network is referred to as an Extranet.

23

24 By joining multiple networks together, it becomes  
25 possible for different organisations to co-operate in the  
26 provision of ATM/kiosk network services. For example,  
27 suppose that a bank which owned a series of conventional  
28 ATMs and an airline which owned a series of ticketing  
29 kiosks chose to co-operate. There exists the potential  
30 for the bank's ATMs to both allow customers to pay for an  
31 airline ticket and to print out that ticket. Similarly,  
32 the airline might offer a limited selection of banking  
33 services, such as balance display, which are compatible  
34 with the functionality of their kiosks.

35

1 Using prior art, the development of such a system would  
2 be complex, particularly due to the different hardware  
3 and capabilities of the bank's ATMs and the airline's  
4 kiosks. Such co-operation between organisations is by no  
5 means impossible at the present time, but is currently  
6 rare due to the complexity and expense required for  
7 implementation.

8  
9 In general, the system provides a means for a plurality  
10 of servers to operate a plurality of ATMs and kiosks  
11 using a plurality of networking means. An example  
12 application would be to allow consumers to purchase eg  
13 cinema, theatre and airline tickets from different  
14 organisations through ATMs positioned at convenient  
15 locations.

16  
17 Typically, the networking means will be the internet, a  
18 corporate intranet or LAN but may be any networking means  
19 or a mixture of networking means.

20  
21 The system comprises a middleware software layer which  
22 extends the function of an underlying operating system  
23 and which in turn provides a single programming interface  
24 for an ATM/kiosk control application to be written to.

25  
26 Figure 4 shows the software architecture of the preferred  
27 implementation of the system. An ATM/kiosk control  
28 application 10 is hosted in a web browser 11 such as  
29 Microsoft®'s Internet Explorer. The application runs on a  
30 computer with a particular operating system, 12, such as  
31 Windows NT®, the functionality of which has been extended  
32 by middleware software 13.

33  
34 The middleware comprises a series of components and  
35 objects, for use by the application, which extend the

1 functionality of the operating system and provide tools  
2 to simplify development of the ATM application.

3  
4 In the preferred implementation all of the system's sub-  
5 systems are implemented as a series of COM components  
6 with an ActiveX<sup>®</sup> interface or as Javabeans<sup>™</sup> with a  
7 scriptable interface. This architecture enables  
8 applications running within Internet Explorer to access  
9 functionality provided by the operating system and the  
10 middleware, including access to hardware.

11  
12 A useful benefit of this implementation is that  
13 applications may be prepared using common authoring tools  
14 and such as Microsoft<sup>®</sup>'s FrontPage<sup>®</sup>, VisualStudio<sup>®</sup>, Visual  
15 Interdev<sup>®</sup> and common development environments such as  
16 Visual Basic<sup>®</sup>, Visual C++<sup>®</sup>, Powerbuilder<sup>®</sup>, Delphi<sup>®</sup> etc.  
17 This means that applications can be prepared with tools  
18 with which developers will be familiar and which, due to  
19 their popularity, provide facilities and support that  
20 would be prohibitively expensive to prepare for a custom  
21 development environment.

22  
23 A further benefit of using browser technology is that  
24 they provide an environment in which software download  
25 can be readily controlled. The application may be held  
26 entirely locally to an ATM/kiosk, entirely on a server or  
27 any compromise between these two extremes. The  
28 application can be downloaded daily if required.

29  
30 The system uses the Windows<sup>®</sup> Open System Architecture  
31 Extensions for Financial Services (WOSA XFS) to support  
32 ATM hardware in a vendor independent manner.

33  
34 The system also uses the Object Linking and Embedding for  
35 Point Of Sale (OPoS) standard for interacting with

1 different hardware systems. This means that applications  
2 can access hardware independent of whether the underlying  
3 hardware supports WOSA XFS or OPOS.

4  
5 The system also supports the PC/SC standard for  
6 smartcards, thereby providing a uniform way of accessing  
7 smartcards.

8  
9 Furthermore, the system also provides support for a  
10 variety of other open standards such as OFX and SNMP and  
11 transaction monitors such as NCR's TOPEND®.

12  
13 Clearly, support for additional standards may readily be  
14 added.

15  
16 The primary subsystems of the middleware software  
17 comprise a series of wizards, device controls, self-  
18 service controls, communications controls and status  
19 monitoring components.

20  
21 The top level components are the wizards, which are a  
22 series of transaction objects that implement common  
23 ATM/kiosk transactions such as dispensing cash, printing  
24 a statement etc. In the preferred embodiment, each is  
25 implemented as an ActiveX® object or a Javabeans™. Whilst  
26 wizards are running, they take control of the function of  
27 the ATM/kiosk. Wizards interface with other controls and  
28 encode all of the top-level control logic.

29  
30 Applications can be built with the system by customising  
31 and combining wizards. Wizards encapsulate all of the  
32 features and functionality required by a particular  
33 transaction or chunk of application. When using ActiveX®.  
34 Wizards receive input via ActiveX® properties and methods  
35 and output their state as a set of ActiveX® events.

1 Alternatively the wizard can be implemented in the same  
2 way as a Javabean™. As a result of this design feature,  
3 the wizard is completely independent of the ATM/kiosk-  
4 user interface.

5  
6 For example, an ATM might have a single button which  
7 dispenses \$10 on demand. A second ATM might implement  
8 more complex controls and display a detailed animation  
9 whilst money is issued. However, the same wizard may be  
10 used to implement both these ATMs. The wizard  
11 encapsulates the essential software logic of the  
12 transaction while allowing the user interface to be  
13 freely defined by script on the browser page.

14  
15 This has several important benefits which will lead to  
16 time and cost savings: firstly, the encapsulated features  
17 within the wizard can be reused between different  
18 applications whilst allowing the different applications  
19 to have totally different look and feel. Secondly, this  
20 allows the user interface to be designed with common web  
21 tools. Thirdly, the user interface may be designed  
22 without any risk of compromising the function of the  
23 wizard. Finally, the user interface may be designed by a  
24 specialist who may not be an expert in the other aspects  
25 of ATM/kiosk software and hardware.

26  
27 An additional important feature of the wizards is that  
28 they are able to interpret the capabilities of the  
29 hardware on which they are run. For example, they may be  
30 able to establish whether a cash dispensing means is  
31 available. One application may then run on a plurality  
32 of different hardware implementations, adapting its  
33 functionality to the capabilities of that hardware.

34



1 This not only allows different hardware implementations  
2 to be incorporated into the same network but allows  
3 distinct networks to be joined into an Extranet.

4

5 The device controls provide hardware independent access  
6 to the special devices on an ATM or kiosk. Each device  
7 control acts as a persistent server that can be  
8 controlled and interrogated by one or more applications  
9 or wizards. A device control abstracts the details of  
10 the hardware underneath it and acts as a complete server  
11 for that device. Applications and wizards interact with  
12 controls through a scriptable ActiveX<sup>®</sup> interface or a  
13 Javabeans<sup>™</sup> interface.

14

15 Some example device controls supported by the system are:

- 16 • Camera
- 17 • Card Reader (motorized, swipe, DIP, smart cards etc.)
- 18 • Cash Acceptor
- 19 • Cash Dispenser
- 20 • Coin Dispenser
- 21 • Depository
- 22 • Doors
- 23 • Encryptor
- 24 • Guide Lights
- 25 • Indicators
- 26 • Journal Printer
- 27 • Keyboards
- 28 • Laser Printers
- 29 • Modems
- 30 • Operator Panel
- 31 • Passbook (including page turn)
- 32 • Pin Pad
- 33 • Receipt Printer

- 1 • Scanner
- 2 • Sensors
- 3 • Signature Capture
- 4 • Statement Printer
- 5 • Touchscreen
- 6 • UPS
- 7 • VendorMode
- 8 • Weighing Scales

9  
10 Multiple applications may be run simultaneously and  
11 device controls are fully concurrent. This is important  
12 as the cycle time of ATMs and kiosk transactions can be  
13 critical. Their design is such that they can be used in  
14 an event-driven manner, with controls reporting their  
15 result (success or failure) via ActiveX<sup>®</sup> or Javabeans<sup>™</sup>  
16 events, or in a procedural manner from within a language  
17 such as C++. In the event-driven mode, applications can  
18 be readily created using browser technology; for example,  
19 readily available web tools which provide appropriate  
20 easy-to-use graphical interfaces can be used to create  
21 event-driven applications.

22  
23 In order to be able to operate asynchronously, all  
24 controls create their own thread, called the event  
25 thread, when first constructed. When an asynchronous  
26 method is called, a command message is sent to the event  
27 thread. The event thread carries out the command and  
28 sends a message back to the main thread on completion:  
29 the completion method causes the appropriate event to be  
30 fired. By implementing commands using the event thread,  
31 the main application thread is free to process other  
32 tasks in parallel. The event thread also ensures that  
33 the device states persist from one application page to  
34 another: although controls on browser pages are being

1 continually created and destroyed, the event thread  
2 remains running and ensures that the connection to the  
3 device is never lost.

4

5 When controls are run in a procedural manner, from a  
6 language such as C++, the controls may be set to a mode  
7 in which events are queued up and delivered to the  
8 application on demand, allowing the application to carry  
9 out other tasks, and return to the event queue at an  
10 appropriate time.

11

12 The self-service controls provide the functionality  
13 necessary for creating self-service applications.  
14 Important self-service controls are described further  
15 below. The communications controls provide access to the  
16 remote host computers. Both the self-service and  
17 communications controls have the same server architecture  
18 as the device controls and all may be executed  
19 asynchronously.

20

21 The status monitoring system monitors the health of the  
22 ATM or Kiosk and sends status and alert signals to an  
23 external monitoring station using SNMP alerts.

24

25 All controls implement a capabilities interface, allowing  
26 an application or wizard to interrogate the capabilities  
27 of the control as well as the device which the control  
28 represents.

29

30 Therefore, not only can different hardware  
31 implementations be integrated into the same network or  
32 Extranet, the applications can dynamically configure the  
33 services they provide depending on the capabilities of  
34 the hardware available on the kiosk.

35

1 As a result of this design, individual software  
2 components can be upgraded without having to change other  
3 aspects of the application. New features can be added  
4 without making the application dependent on those  
5 features.

6  
7 Furthermore, hardware and networking components may be  
8 upgraded or altered step by step. Due to the modular  
9 nature of the system and its customisability, a plurality  
10 of communications and hardware implementations may be  
11 used at once. This means that an organisation which runs  
12 an ATM/kiosk network might use its legacy communications  
13 and hardware implementations, perhaps concurrently with  
14 Internet/Intranet support. This means that ATM networks  
15 may be implemented and altered step-wise.

16  
17 Such upgrades are particularly easy when using the Open  
18 Financial Exchange (OFX) architecture. The middleware  
19 software implements a single OFX Control which may  
20 interface with an OFX server by any networking means.  
21 The OFX server may also interface with a host by any  
22 networking means. Once this architecture is implemented,  
23 the resulting network topology may be readily altered,  
24 making this an easy migration path for existing networks  
25 to use this system.

26  
27 A further implication of the design of the controls is  
28 that they can run on an ATM/kiosk even when actual  
29 hardware devices are not present. This allows the  
30 applications to be started up and run, for example for  
31 development and test purposes, without requiring  
32 particular hardware. When the application requests the  
33 capabilities of a particular control, the control will  
34 reply that the device is not present and that the  
35 capabilities are null. Therefore it is possible to

1 create and test application on, for example, a PC. In  
2 this situation, the PC will behave like an ATM/kiosk in  
3 its interactions with the application.

4

5 An ignore mode is also provided wherein particular  
6 controls will return "success" for every command. This  
7 allows the application to use generic code which does not  
8 need to test whether the device is present at each step,  
9 simplifying the code that needs to be written when  
10 creating an application to cope with various hardware  
11 capabilities.

12

13 An HTML-based application is also provided with the  
14 system for testing device controls. This application  
15 allows the operator to select a subset of the devices for  
16 testing. For each device, two test sequences are  
17 defined: one requires operator interaction (e.g.  
18 entering/removing a card) and one requires no operator  
19 interaction. When the latter is selected, the  
20 interaction-free test sequences will be repetitively run  
21 for the selected devices, allowing applications provided  
22 using this system to be easily stress tested. Complete  
23 tests including operator interaction may also be  
24 selected. Testing is automated and therefore as  
25 reproducible as possible.

26

27 All controls include a security mechanism. This  
28 mechanism allows the methods of the various controls to  
29 be enabled and disabled. This is particularly important  
30 in an Extranet environment when applications of differing  
31 abilities run on a given kiosk or ATM. For example, if a  
32 bank operating a network of ATMs allowed an airline to  
33 dispense tickets through its ATMs by way of an Extranet,  
34 it would wish to disallow the airline's application from  
35 dispensing cash.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35

This security mechanism is implemented by a key passing technique as follows:

The middleware software contains a security control which allows the current security configuration of an ATM or kiosk to be set. Using the security control, the owner of the ATM or kiosk can specify details of the security configuration (i.e. which methods of a control are allowed and disallowed). Applications identify themselves to the security control via a digital certificate which sets the security configuration as specified by the ATM/kiosk owner. If the application attempts to call a disallowed method of control, a trap is generated, transferring control to the ATM/kiosk owner's application.

An important benefit of the system is that it may readily be used to provide internet based e-commerce facilities through ATMs and kiosks, not only allowing e-commerce facilities to be used by a larger consumer base but also enabling e-commerce which requires expensive or high-security hardware facilities such as cash dispensers or identity verification means that cannot readily be provided on privately owned PCs and web-browsers.

To help enable this, the system provides a Site-Minder control which allows existing web sites to be safely delivered via ATMs and kiosks. This control provides several important features. For example, it monitors the URL of each page of the web-site being delivered and allows or disallows the page according to a rules database. This stops the user from straying into other web-sites or web-pages that are not normally part of the purpose of the ATM/kiosk. The control allows each page

1 to be given a customised time-out which is important as  
2 web sites are normally designed for use at home and have  
3 different (longer) time-outs than would be appropriate  
4 for public ATMs/kiosks. Web pages may be navigated using  
5 a touch sensitive screen, making them intuitive and easy  
6 to use. The control can also magnify small features on a  
7 web page (such as hypertext links and images with links)  
8 This magnification can be toggled on and off by the user,  
9 thereby animating the hypertext link. This is beneficial  
10 firstly because it makes it easier for the user to see  
11 where the link is and secondly because it becomes easier  
12 for the user to select the link when it is in its  
13 magnified state.

14  
15 An additional feature provided by the system for use with  
16 ATMs/kiosks with touchscreens is a "softkeyboard" wherein  
17 a keyboard is displayed on the touch screen and contact  
18 with the displayed keyboard is interpreted by the system  
19 like keystrokes on a real keyboard, thereby removing the  
20 need for a physical keyboard to be provided.

21  
22 One problem commonly faced by web designers is that  
23 objects placed on a web page are destroyed when the page  
24 is changed. A useful benefit of the middleware is that  
25 the ActiveX<sup>®</sup> hook idea solves this problem - underlying  
26 objects remain persistent while lightweight hooks on each  
27 page access the object.

28  
29 Lack of persistence also leads to problems for the  
30 application developer in storing application-wide data.  
31 A solution to this problem is provided by a scratchpad  
32 control which has a persistent object at its core and  
33 allows the application to store and retrieve data at any  
34 time. This control supports the Vbscript variant type,  
35 allowing all types of data to be stored and retrieved.

1 Furthermore, this control allows data to be shared  
2 between multiple applications, marking it as shared.

3  
4 A related problem when implementing web-based ATM  
5 applications relates to events which must be dealt with  
6 immediately, no matter when the event occurs. For  
7 instance, if a safe door is opened, an application may  
8 need to shut down immediately. This would not be easy to  
9 implement in a web-based environment as every page would  
10 have to contain some code to handle the event. This  
11 problem can be solved in the system by operating a  
12 second, invisible frame alongside the main application  
13 frame. The invisible frame contains all the device  
14 controls needed to detect the events that must be reacted  
15 to. This frame may then take control, perhaps closing  
16 down the main frame.

17  
18 Error handling in traditional ATM applications is  
19 difficult. Components may return a large number of error  
20 cases, resulting in complex code.

21  
22 The middleware software separates the responses it sends  
23 to the application into "good responses" and error  
24 responses. Most commands have a single good response and  
25 all errors are mapped to a single error response,  
26 although some may have a plurality of good responses.  
27 Good responses allow the application to continue. When  
28 an error response is returned, the current transaction  
29 flow is normally aborted and control flow jumps out of  
30 the normal flow process to handle the error situation.  
31 The application can then interrogate the control to  
32 determine the exact cause of the error.

33  
34 A benefit of this approach is that normal flow is not  
35 cluttered by handlers for each of the error cases which



1 can occur. Control may be transferred to generic error  
2 handlers which can either recover from the error or abort  
3 the transaction completely, perhaps even rebooting the  
4 ATM/kiosk. Application code can therefore remain as  
5 clear and concise as possible whilst encouraging the  
6 application developer to handle all error cases by  
7 calling an error handler. In the development  
8 environment, fatal errors result in a message box being  
9 displayed. A single type of event, DeviceError, is  
10 generated when there is some kind of hardware failure,  
11 allowing error handling for hardware failure to be  
12 encapsulated rather than scattered over many error  
13 handlers.

14  
15 The system requires applications to interact with it in a  
16 well defined way. Even small transgressions are detected  
17 and error responses generated; when this happens, the  
18 current environment is abandoned and the application is  
19 terminated.

20  
21 This is based on the well known software engineering  
22 approach of assertion; however, the system's assertion  
23 differs from common practice by asserting absolutely all  
24 disallowed cases, whether serious or not. As a result of  
25 this strategy of escalating errors to maximum  
26 seriousness, errors are found earlier at development time  
27 or at system test time and never allowed to reach a live  
28 environment. Although there is a risk of the application  
29 reporting a fatal error in the field for a relatively  
30 minor problem, this strategy achieves a particularly high  
31 level of robustness in comparison to prior art software  
32 applications.

33  
34 An additional error-handling feature is provided by the  
35 way in which the system deals with tracing. In software

1 engineering, tracing is typically enabled only when a  
2 problem is suspected; however, this can affect the  
3 dynamics of a program, making it harder to find bugs.  
4 This is a particularly substantial problem when dealing  
5 with time-critical ATM/kiosk applications. However, if  
6 conventional tracing was simply always enabled throughout  
7 both development and operation of the ATM/kiosk, there  
8 would be both performance problems due to, for example,  
9 the time spent writing to a hard drive and large quantity  
10 of disk space required to store the large number of trace  
11 events that will typically be produced.

12  
13 The middleware software provides a trace control which  
14 records all trace events of the application and  
15 underlying middleware and is always enabled. Performance  
16 problems are dealt with by writing trace data to memory  
17 and writing to disk only when the ATM/kiosk is idle.  
18 Cash-dispensing machines and kiosks go through an idle  
19 cycle between two users which provides sufficient time to  
20 write to disk, even when people are queuing at the  
21 machine. Disk space problems are eliminated by using a  
22 ring buffer of fixed file size, allocated at boot-up and  
23 constant in size throughout operation. When the buffer  
24 is full, the oldest data is overwritten, thereby leaving  
25 a continual record of the most recent events.

26  
27 As a result of this tracing strategy it is much easier to  
28 understand one-off or rare problems, which is not easily  
29 done when tracing is enabled only once a problem has been  
30 reported.

31  
32 Furthermore, some ATM/kiosk vendors provide a limited  
33 amount of non-volatile RAM. When this is provided, the  
34 trace control writes the most recent trace information to  
35 this RAM in a ring buffer fashion. As this is very

1 quick, it does not produce any performance problems.  
2 However, if the ATM/kiosk freezes up or crashes, the RAM  
3 contains the trace of what happened immediately before.  
4

5 In addition to the traditional way that ActiveX® fires  
6 events to the container, the device and self-service  
7 controls are able to queue up events and return them one  
8 by one when requested. This allows C++ applications to  
9 be written in a procedural fashion rather than simply in  
10 an event driven fashion. By queuing up these events and  
11 delivering them to the application only on demand, the  
12 system allows procedural code to be written and makes it  
13 easier to develop and maintain the complex logic required  
14 in self-service applications.  
15

16 Important self-service controls are described below:  
17

- 18 • Watchdog control: runs in a separate Windows NT®  
19 process and reboots the ATM/kiosk if the application  
20 crashes. This is achieved by regularly polling the  
21 application to check that it is functioning correctly.  
22 This control can also be used to daily reboot the  
23 ATM/kiosk. The watchdog can monitor multiple  
24 applications on a single ATM.
- 25 • System Escape control: used to reboot the ATM/kiosk.  
26 Exits in a customisable manner. This control ensures  
27 that cached data (eg in the DataCollect control and the  
28 Trace control) is flushed to disk before rebooting.
- 29 • DataCollect control: allows application to collect raw  
30 data for statistical purposes. It logs and timestamps  
31 the various events. As with the Trace control, it logs  
32 to memory and then stores on hard disk only when the  
33 ATM/kiosk is idle due to the time required to write to  
34 the hard disk. Storage by this control is of a fixed  
35 size allocated at start-up and remaining constant

1 throughout operation. Storage is in the form of a ring  
2 buffer. Typically, the collected data would be  
3 exported to a remote location for analysis.

- 4 • Trace control: described above.
- 5 • Scratchpad control: described above.
- 6 • Supervisor application: run simultaneously as a  
7 separate application. This means that on an ATM/kiosk  
8 with a rear screen, the operator can interact with the  
9 ATM/kiosk without taking the machine offline. It  
10 allows the operator to access statistics etc. while the  
11 machine is still being used. Alternatively, the  
12 machine may be taken off-line for intrusive  
13 maintenance. In this case, the supervisor application  
14 provides an off-line mode with a limited subset of the  
15 on-line features.
- 16 • Security control: described above.
- 17 • Registry control: allows Windows NT<sup>®</sup> registry to be  
18 manipulated by the application.
- 19 • DirectoryTree control.
- 20 • Application Launcher control.
- 21 • INI file control: allows Windows<sup>®</sup> INI files to be read  
22 from the browser.
- 23 • Timed FTP. This allows statistics files and trace files  
24 to be sent via the FTP mechanism on a timed basis to an  
25 offsite location. (eg daily or weekly).
- 26 • Key capture control: allows special Windows<sup>®</sup> key  
27 combinations such as ctrl-alt-del and alt-tab to be  
28 captured where a full PC keyboard is provided.
- 29 • Popup suppression control. Monitors and captures popup  
30 windows originating from the operating system. This  
31 makes it easier to allow software components from other  
32 vendors to be used in self-service applications. Most  
33 third-party software is not intended for self-service  
34 applications and expects to be able to interact with

1 the user through popup windows. This is unacceptable  
2 in a self-service environment where the main  
3 application must have a complete monopoly over the user  
4 dialog. This control alleviates this problem by  
5 monitoring popups and rapidly executing a pre-  
6 determined sequence of tasks, for example hiding the  
7 popup and pressing the OK button.

- 8 • Global config file control. Allows configuration data  
9 for ATM networks to be centrally held in a single  
10 distributable file. Each ATM/kiosk can query this  
11 control to retrieve the configuration data which is  
12 specific for that ATM/kiosk. This allows variation  
13 between individual ATMs/kiosks to be handled in a  
14 global way.
- 15 • Telephony control. Allows modems and telephone handsets  
16 to be integrated.
- 17 • SSMS control. Allows software to be downloaded and  
18 installed in a controlled manner. This control checks  
19 for installation failures and allows the system to  
20 recover to a well defined state.
- 21 • Screensaver control. This control allows the  
22 application to jump to a defined web page if the user  
23 has been inactive for more than a pre-determined time.
- 24 • Multiple language control. This control allows the  
25 language on a web page to be dynamically modified. It  
26 does this by retrieving text strings and graphics from  
27 a database on the kiosk. This means that the user may  
28 change languages from any browser page - and therefore  
29 at any stage of the application.
- 30 • Clock synch control. This allows the application to  
31 synchronize its clock with a server clock, taking into  
32 account possible differences in timezone between kiosk  
33 and server and taking into account the possibility of  
34 large timelags for communication between the kiosk and  
35 the server.

1 Use of the self-service controls plus additional features  
2 of the system and underlying operating system allow  
3 ATMs/kiosks to be managed from a remote location. For  
4 example, the system supports:

- 5 • Daily software downloads from a remote web server.
- 6 • Daily reboot and system check.
- 7 • Daily FTP of statistics data to a remote monitoring  
8 station.
- 9 • Daily FTP of trace data to a remote monitoring system.
- 10 • Regular health checks of the kiosk (typically every 5  
11 minutes).
- 12 • Sending a regular "heartbeat" message to a remote  
13 monitoring station. Monitoring of this message allows  
14 the fact that the device is continually functioning to  
15 be monitored.
- 16 • Allowing direct secure access to the kiosk over a  
17 network, for example the Internet, from a remote  
18 location.
- 19 • Allowing software maintenance over a network, for  
20 example the Internet, from a remote location.
- 21 • Allowing manual reboot of the kiosk over a network, for  
22 example the Internet, from a remote location.

23  
24 Although hardware is accessed via the WOSA XFS standard,  
25 which assigns a different number to each command, the  
26 controls have differently named methods and events  
27 associated with each operation, making application  
28 development easier. WOSA commands may typically generate  
29 30-50 events. This wastes time for the application  
30 developer and increases the possibilities of error. The  
31 middleware reduces the set of possible outcomes to a  
32 small number of clearly named completion events, making  
33 it easier for the application developer to write reliable  
34 code quickly. Outcomes which can only happen if there is

1 a bug in the application cause fatal errors to be  
2 triggered.

3  
4 The system automatically opens a WOSA XFS session when a  
5 device control is first used; there is therefore no need  
6 to manually call an Open method. WOSA sessions are  
7 maintained between pages through the use of event  
8 threads, described above.

9  
10 All WOSA XFS methods require a timeout to be provided;  
11 however, this is not appropriate or meaningful for the  
12 majority of commands in this application. The middleware  
13 requires a timeout to be supplied only where it is  
14 meaningful to do so. WOSA also allows cancel commands to  
15 be sent after any other command. Not all ATM functions  
16 can really be cancelled and the middleware only provides  
17 cancel commands where cancellation can actually be  
18 achieved. The request IDs returned by WOSA for each  
19 asynchronous operation are abstracted out by the  
20 middleware. WOSA is accessed only by the middleware and  
21 not directly by the application.

22  
23 Clearly the preferred embodiment described above may  
24 readily be adapted to operate with any operating system  
25 or component system.

26  
27 Further modifications and improvements may be  
28 incorporated without departing from the scope of the  
29 invention herein intended.

30